EECS 370

Set-associative Caches

Announcements

- Extended OH today
- Lab 10 meets Fr/M
- Full P3 due tonight

Poll: **What storage is needed per block?**

Class Problem—Storage overhead

- Consider the following cache:
  32-bit memory addresses, byte addressable, 64KB cache
  64B cache block size, write-allocate, write-back, *fully associative*

  This cache will need 512 kilobits for the data area (64 kilobytes times 8 bits per byte). Note that in this context, 1 kilobyte = 1024 bytes (NOT 1000 bytes!) Besides the actual cached data, this cache will need other storage. Consider tags, valid bits, dirty bits, bits to keep track of LRU, and anything else that you think is necessary.
- How many additional bits (not counting the data) will be needed to implement this cache ?

Poll: What storage is needed per block?

Class Problem—Storage overhead

- Consider the following cache:
  32-bit memory addresses, byte addressable, 64KB cache
  64B cache block size, write-allocate, write-back, *fully associative*

  This cache will need 512 kilobits for the data area (64 kilobytes times 8 bits per byte). Note that in this context, 1 kilobyte = 1024 bytes (NOT 1000 bytes!) Besides the actual cached data, this cache will need other storage. Consider tags, valid bits, dirty bits, bits to keep track of LRU, and anything else that you think is necessary.
- How many additional bits (not counting the data) will be needed to implement this cache ?

Tag bits = 32 – log(64) = 26 bits

#lines = 64KB/64B = 1024

LRU = log(1024) = 10 bits

1 valid bit, 1 dirty bit

Poll

Class Problem—Analyze performance

- Suppose that accessing a cache takes 10ns while accessing main memory in case of cache-miss takes 100ns. What is the average memory access time if the cache hit rate is 97%?

- To improve performance, the cache size is increased. It is determined that this will increase the hit rate by 1%, but it will also increase the time for accessing the cache by 2ns. Will this improve the overall average memory access time?

Class Problem—Analyze performance

- Suppose that accessing a cache takes 10ns while accessing main memory in case of cache-miss takes 100ns. What is the average memory access time if the cache hit rate is 97%?

  AMAT = 10 + (1 - 0.97)*100 = 13 ns

- To improve performance, the cache size is increased. It is determined that this will increase the hit rate by 1%, but it will also increase the time for accessing the cache by 2ns. Will this improve the overall average memory access time?

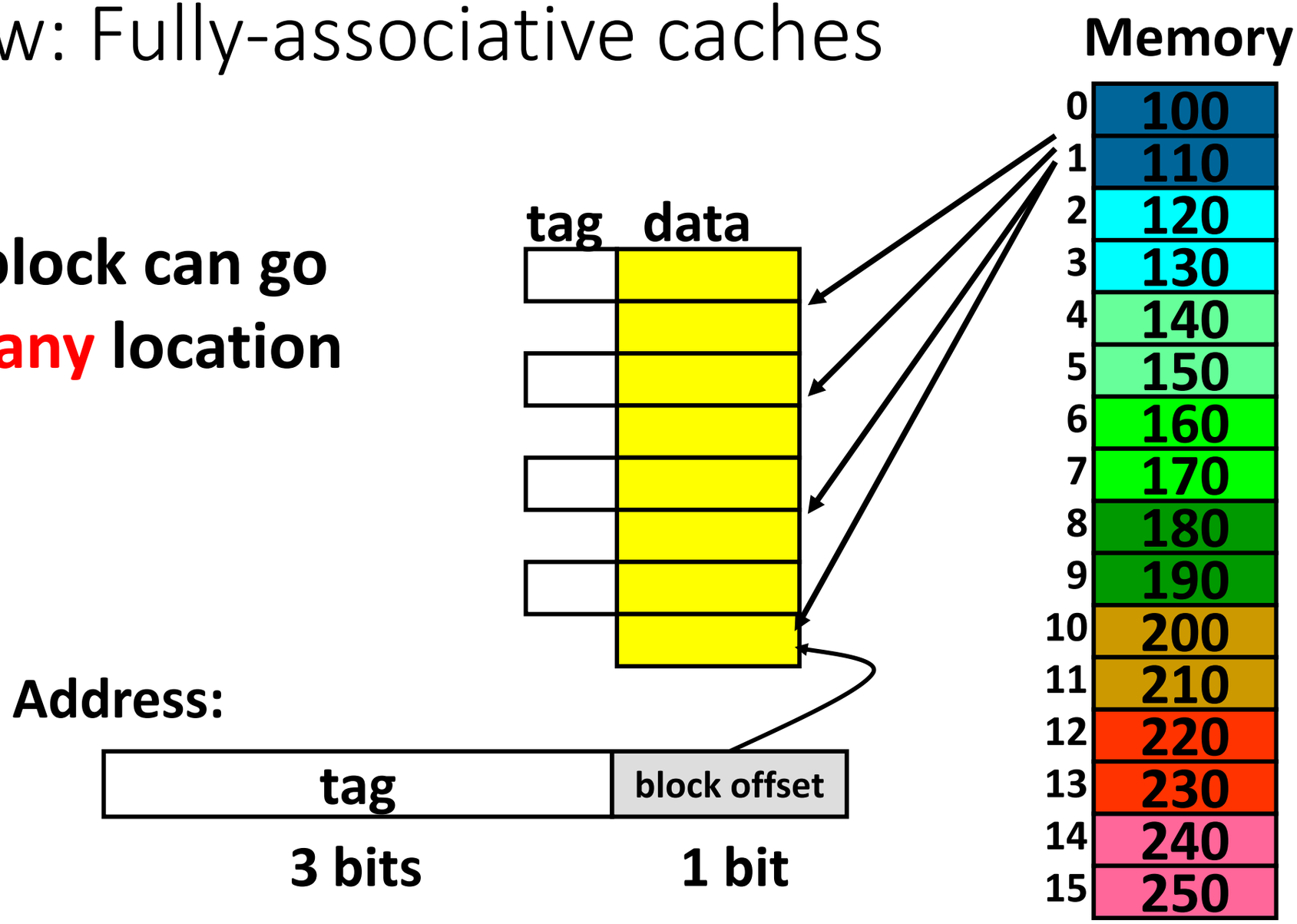
  AMAT = 12 + (1 - 0.98)*100 = 14 ns

Agenda

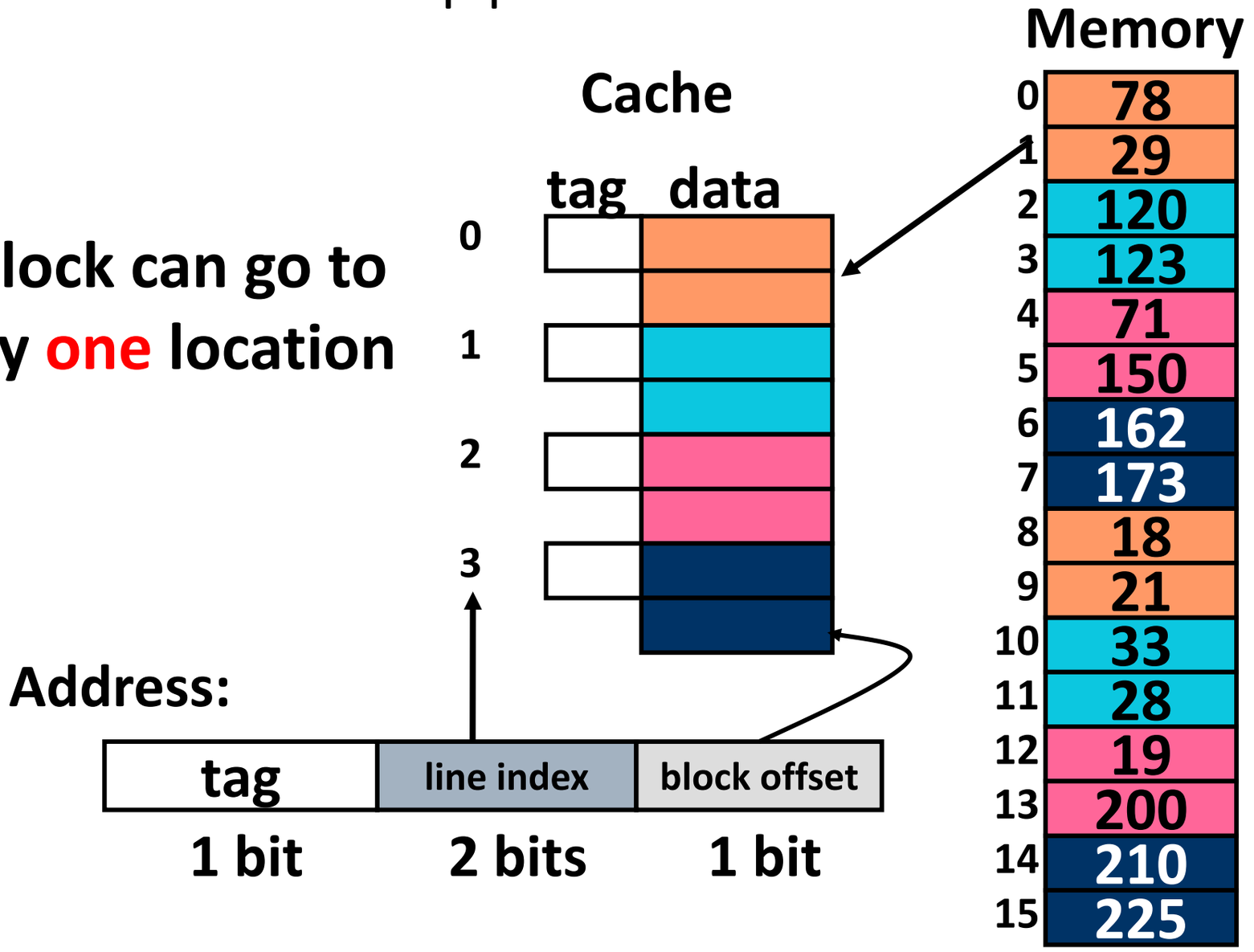
- **Set-associativity overview**
- Example
- Class problem
- Integrating caches into our processor

Review: Fully-associative caches

A block can go to any location



Review: Direct-mapped caches



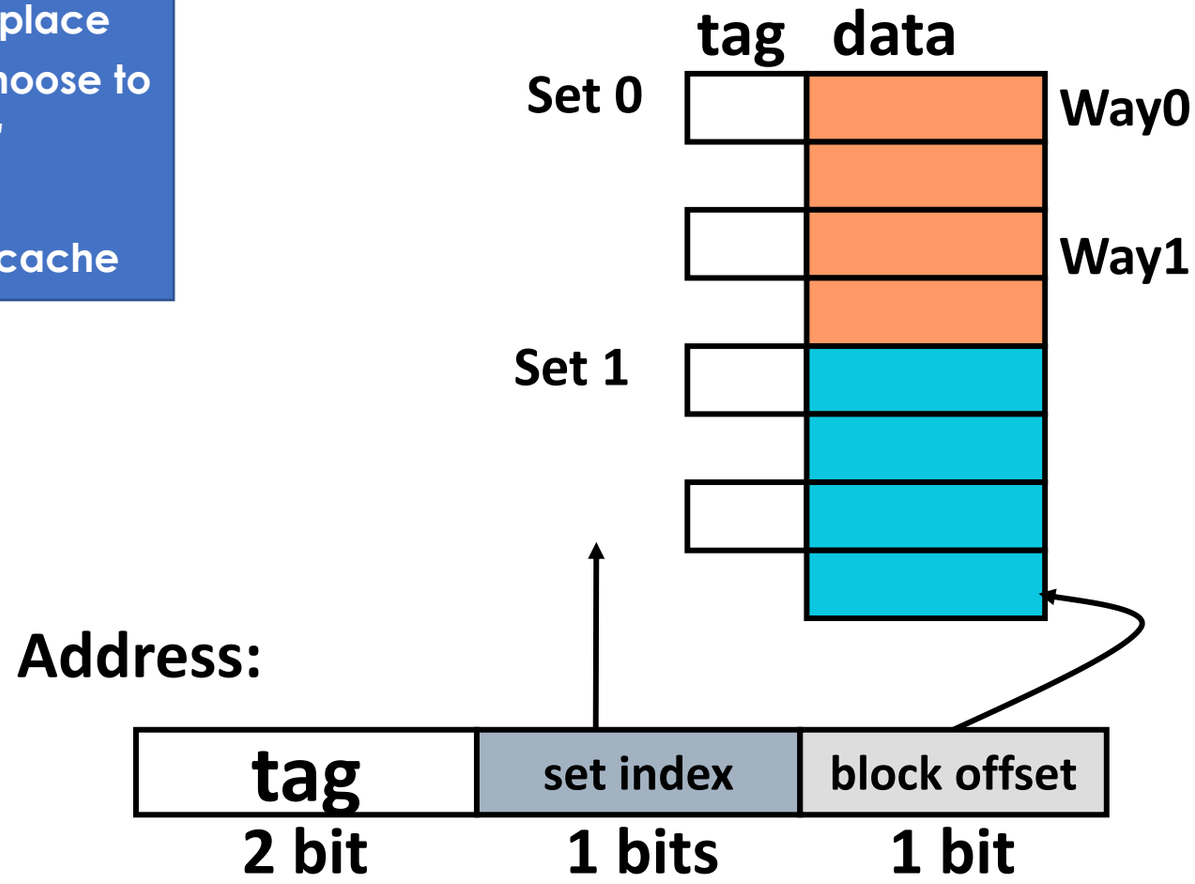
Set-associative caches

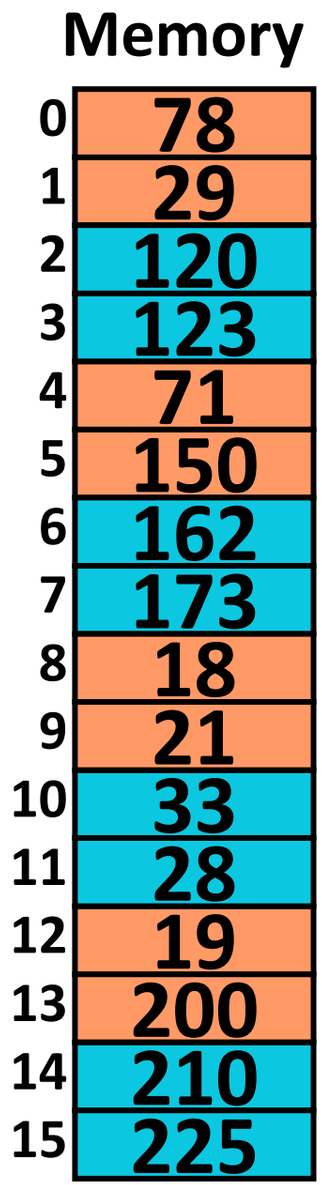
- Fully-associative & direct mapped are two extremes:
  - Slow & full placement flexibility vs fast & no placement flexibility
  - Can we do something in the middle?
- Set associative caches:
  - Partition memory into regions
    - like direct mapped but fewer partitions
  - Associate a region to a set of cache lines
    - Check tags for all lines in a set to determine a HIT
- Treat each set like a small fully associative cache
  - LRU (or LRU-like) policy generally used

Set-associative cache

"Way" means "place hardware can choose to put data"

This is a 2-way cache





Calculating all the bit sizes

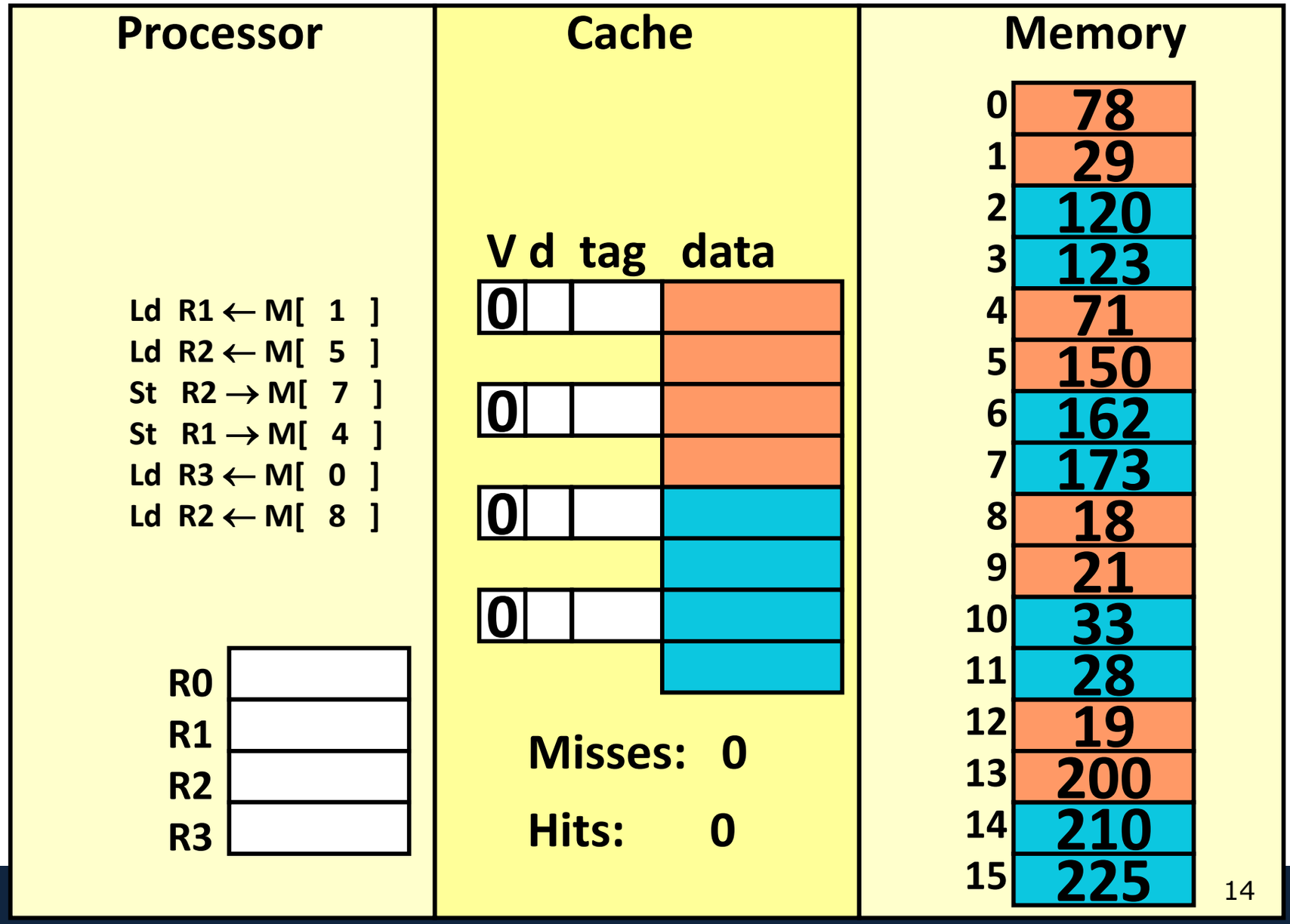
Poll: How many sets are in a direct mapped cache with N cache lines?

- For a set-associative cache:
  - # block offset bits = $\log_2$(block size)
  - # set index bits = $\log_2$(# of sets)
  - # tag bits = rest of address bits
- Fully-associative
  - Special case where (# sets) = 1
- Direct-mapped:
  - Special case where (# sets) = (# cache lines)

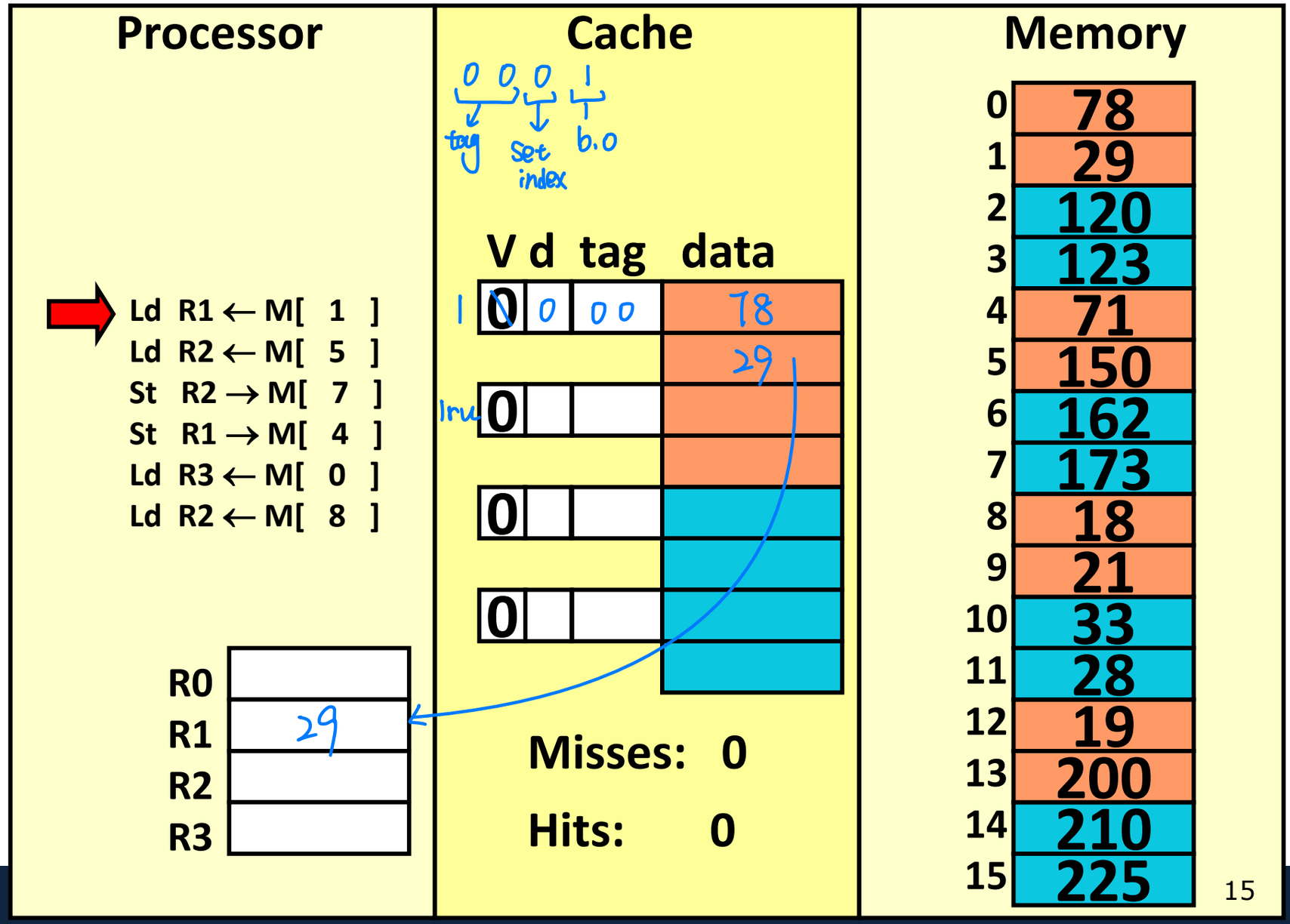
Agenda

- Set-associativity overview
- **Example**
- Class problem
- Integrating caches into our processor

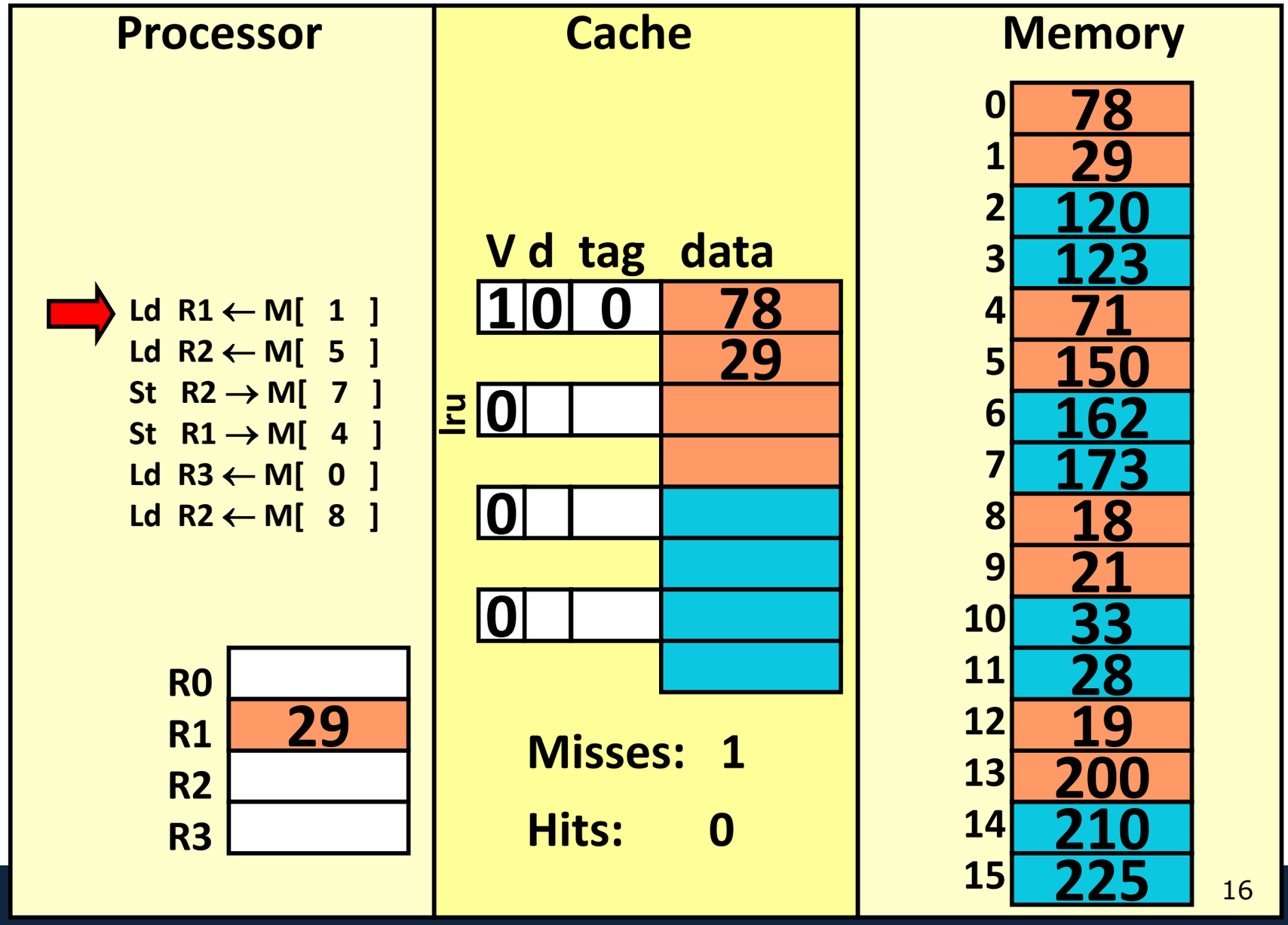
Set-associative cache example (Write-back, write allocate)



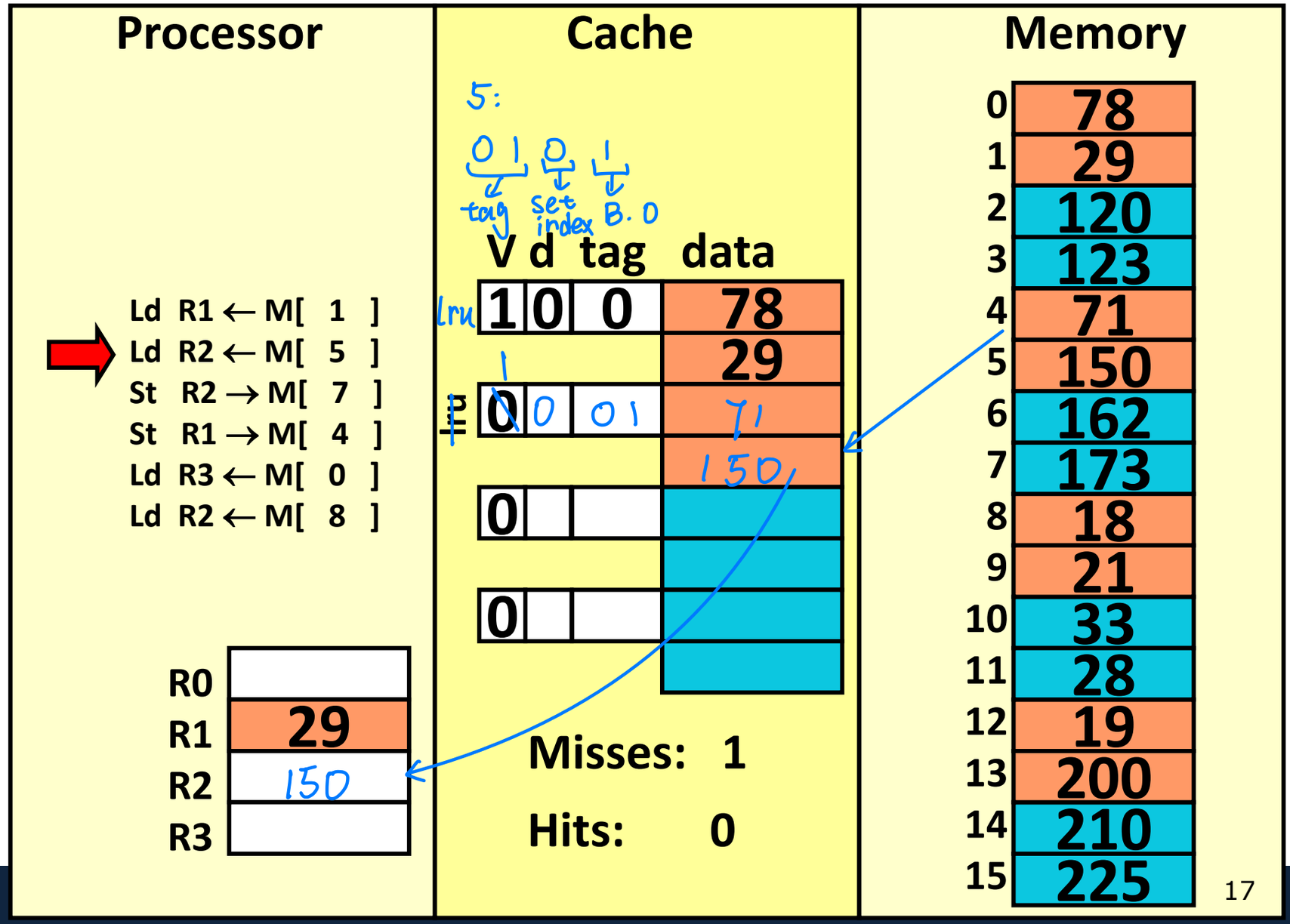
Set-associative cache (REF 1)



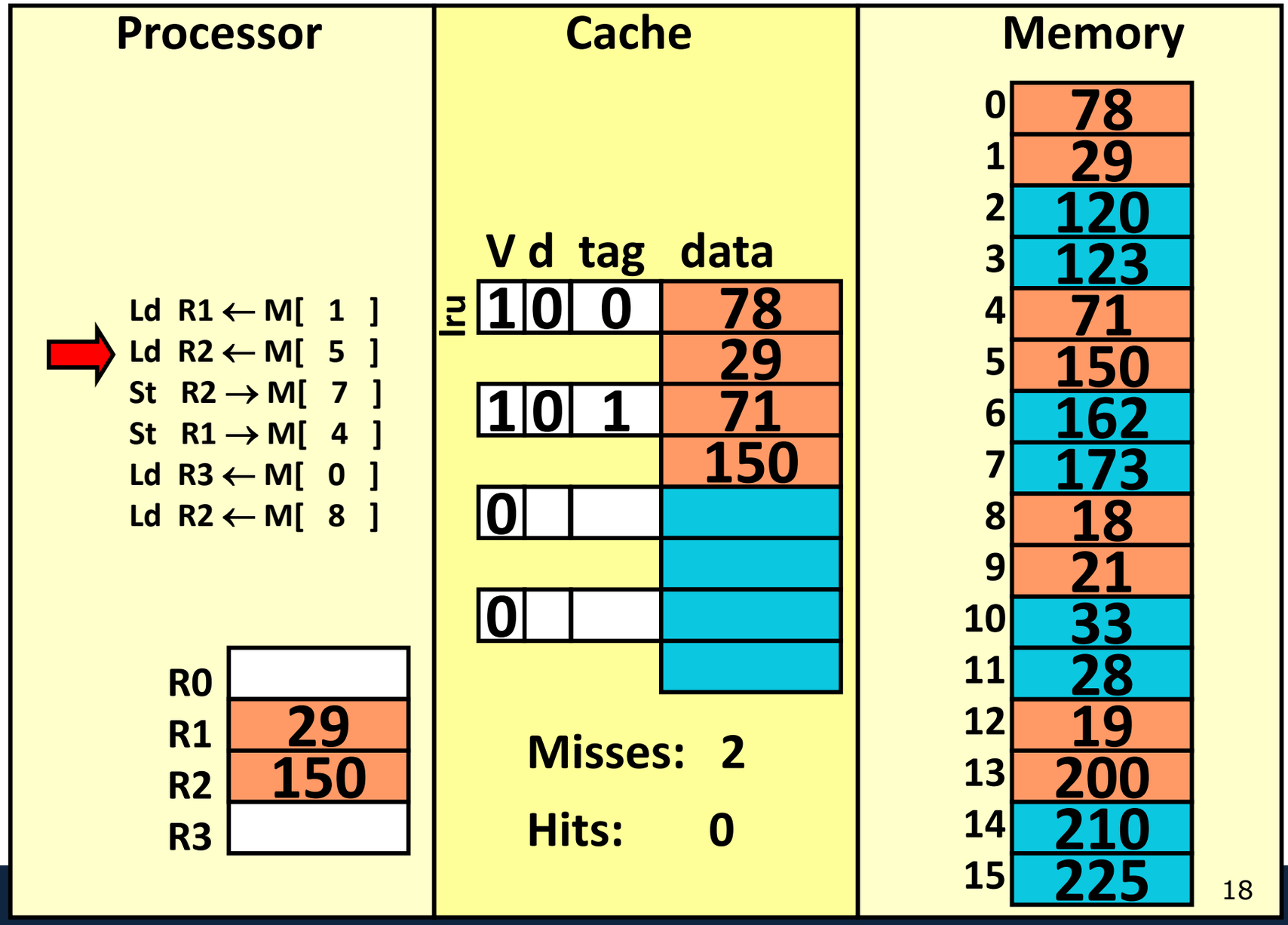
Set-associative cache (REF 1)



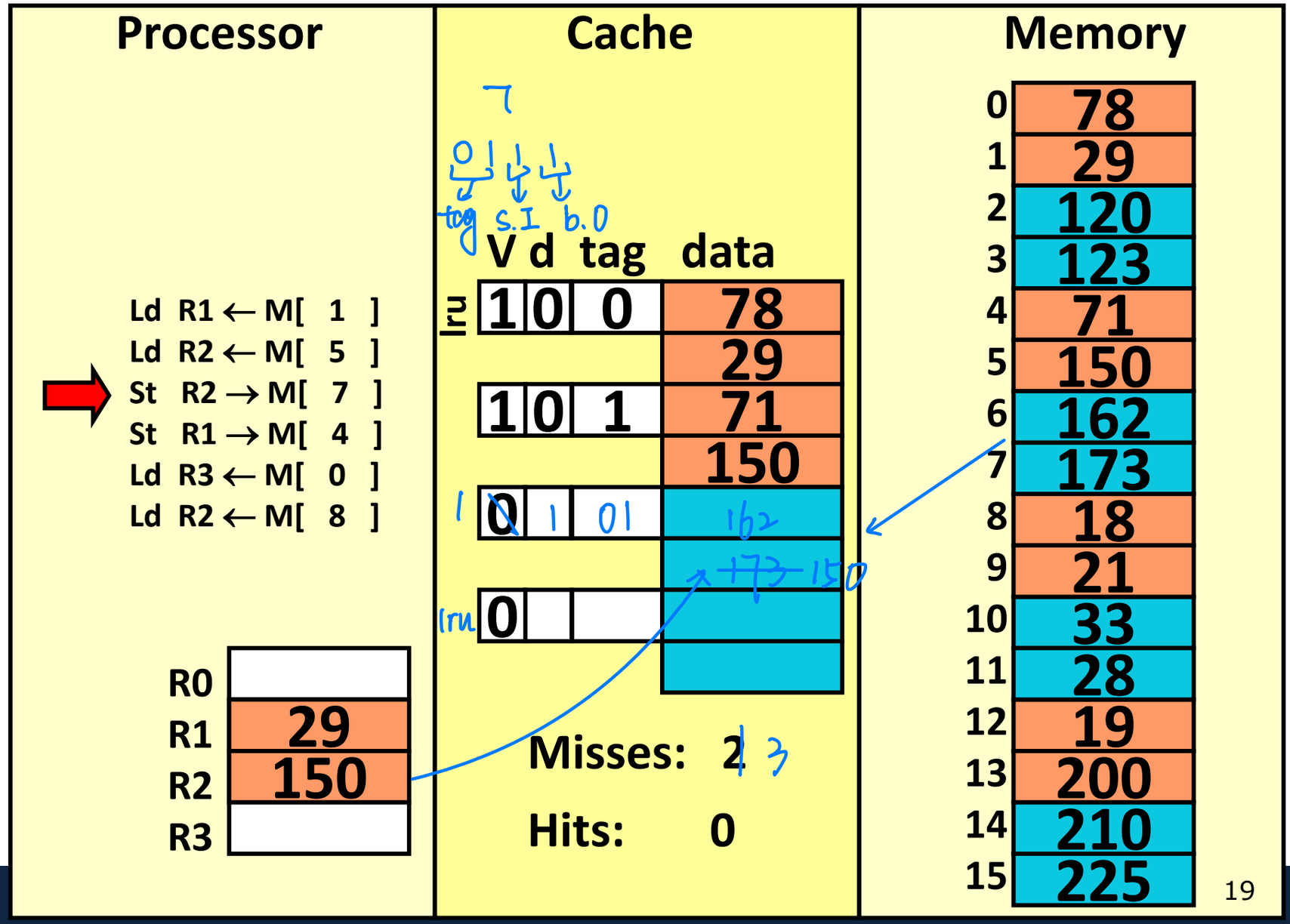
Set-associative cache (REF 2)



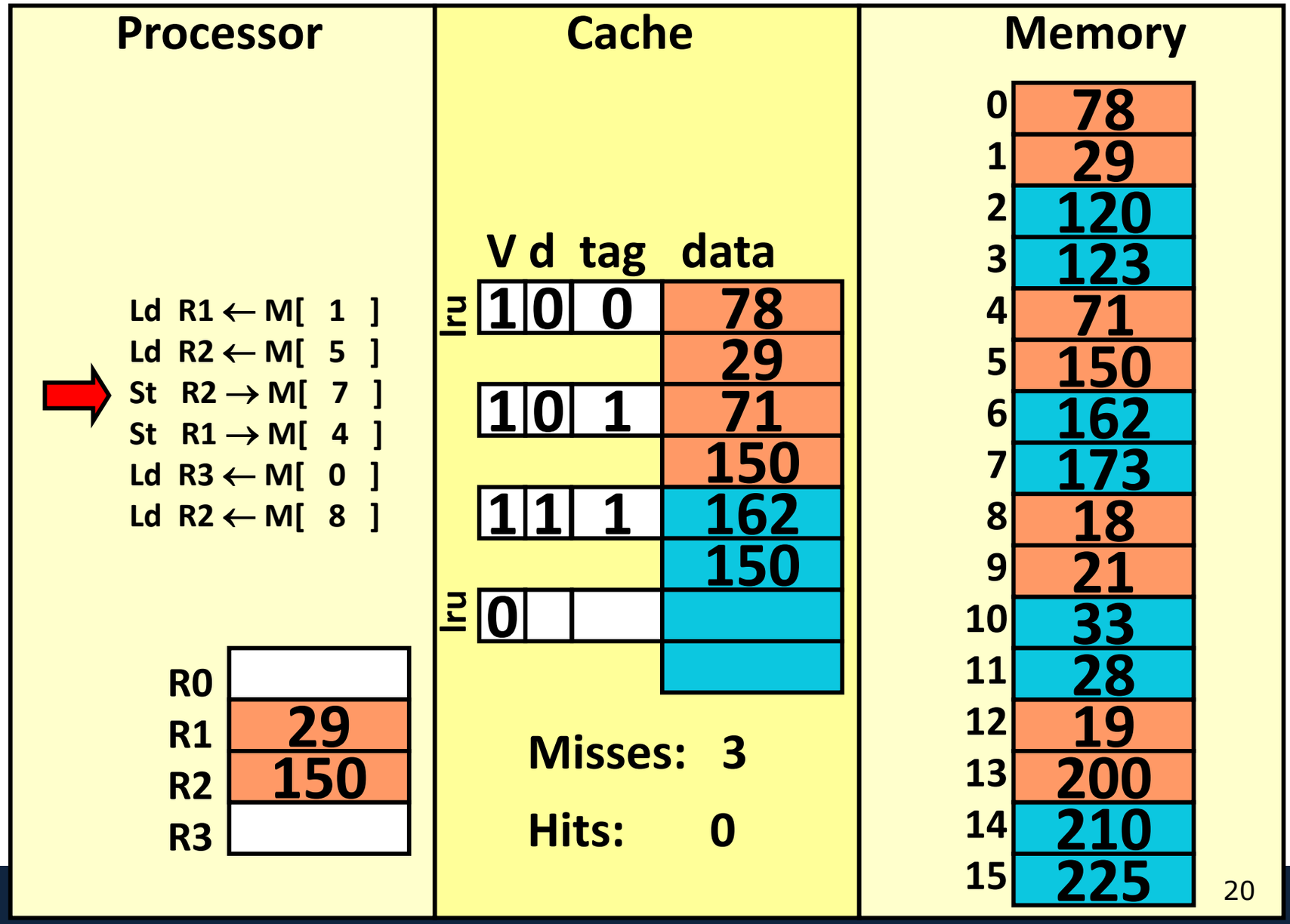
Set-associative cache (REF 2)



Set-associative cache (REF 3)

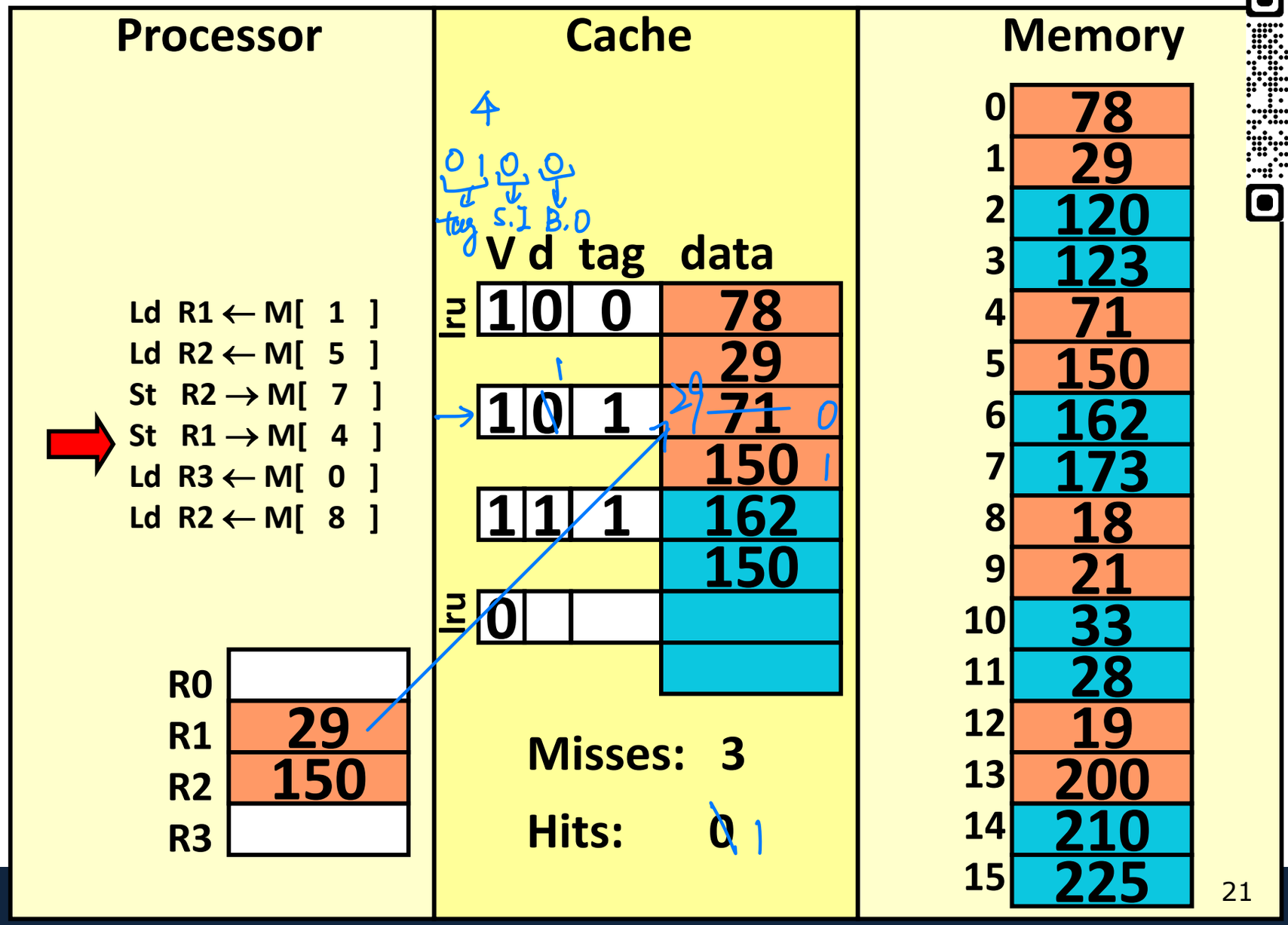


Set-associative cache (REF 3)



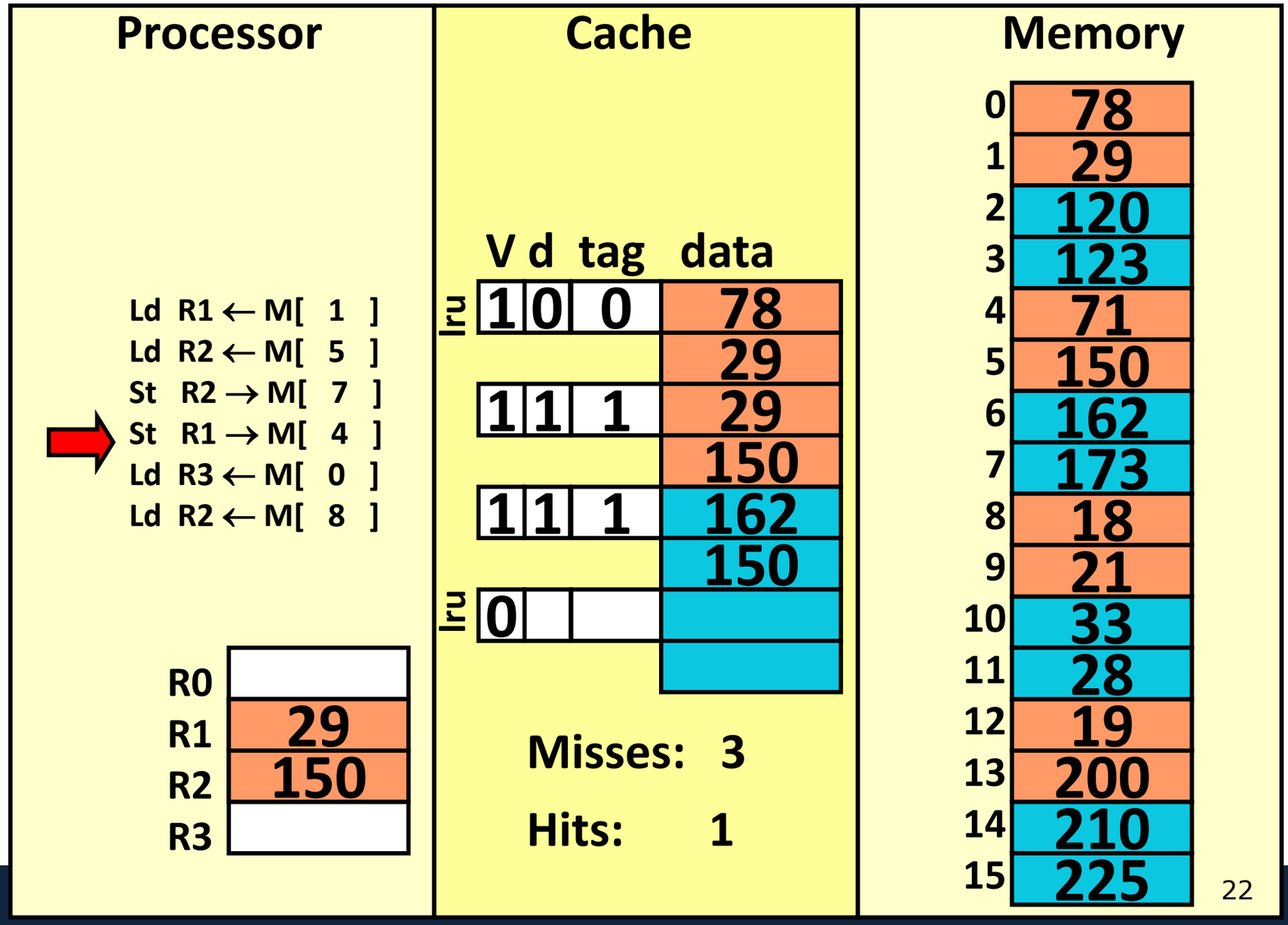
Set-associative cache (REF 4)

Poll: Finish the remaining references

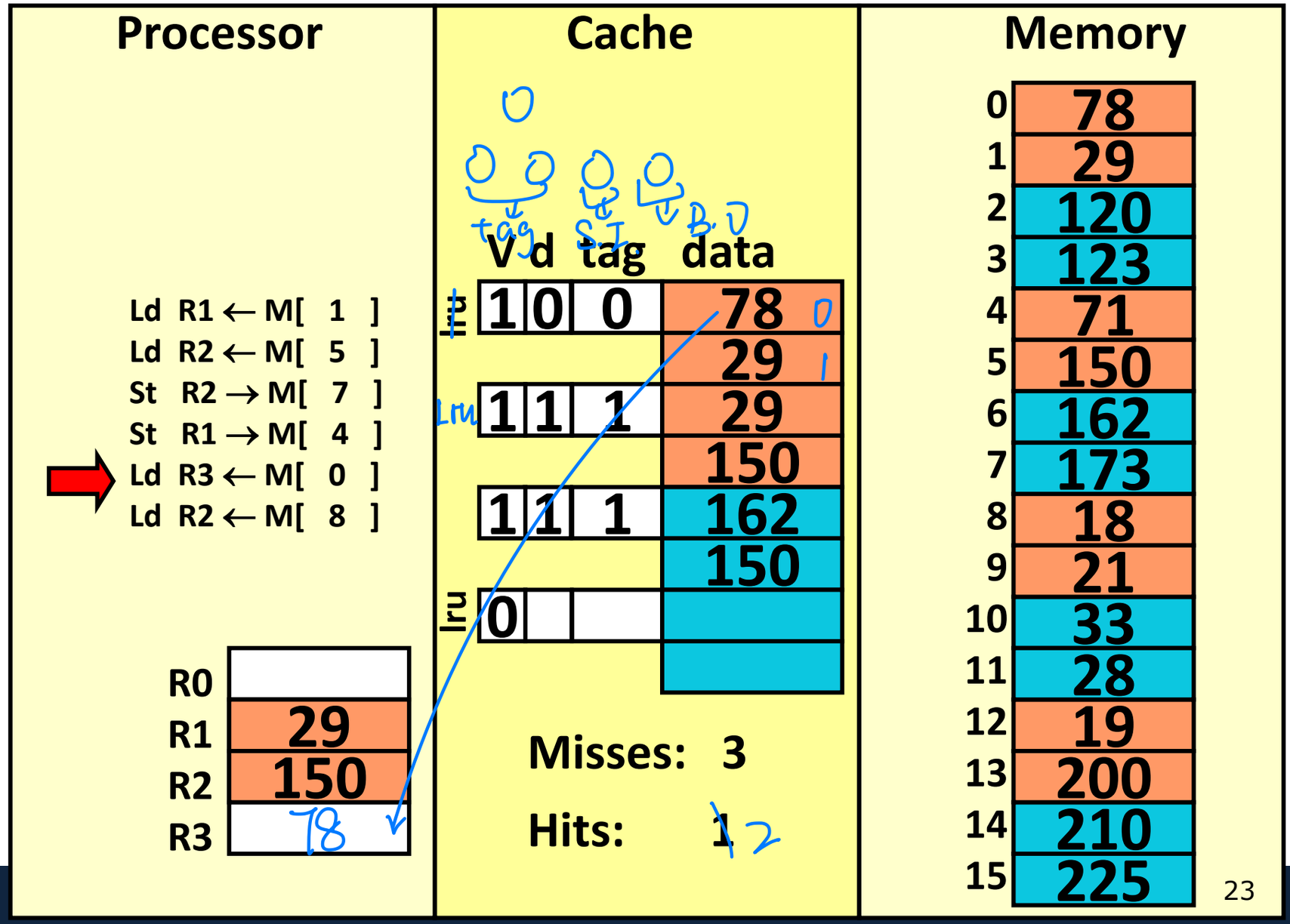


https://bit.ly/3x7nKpP

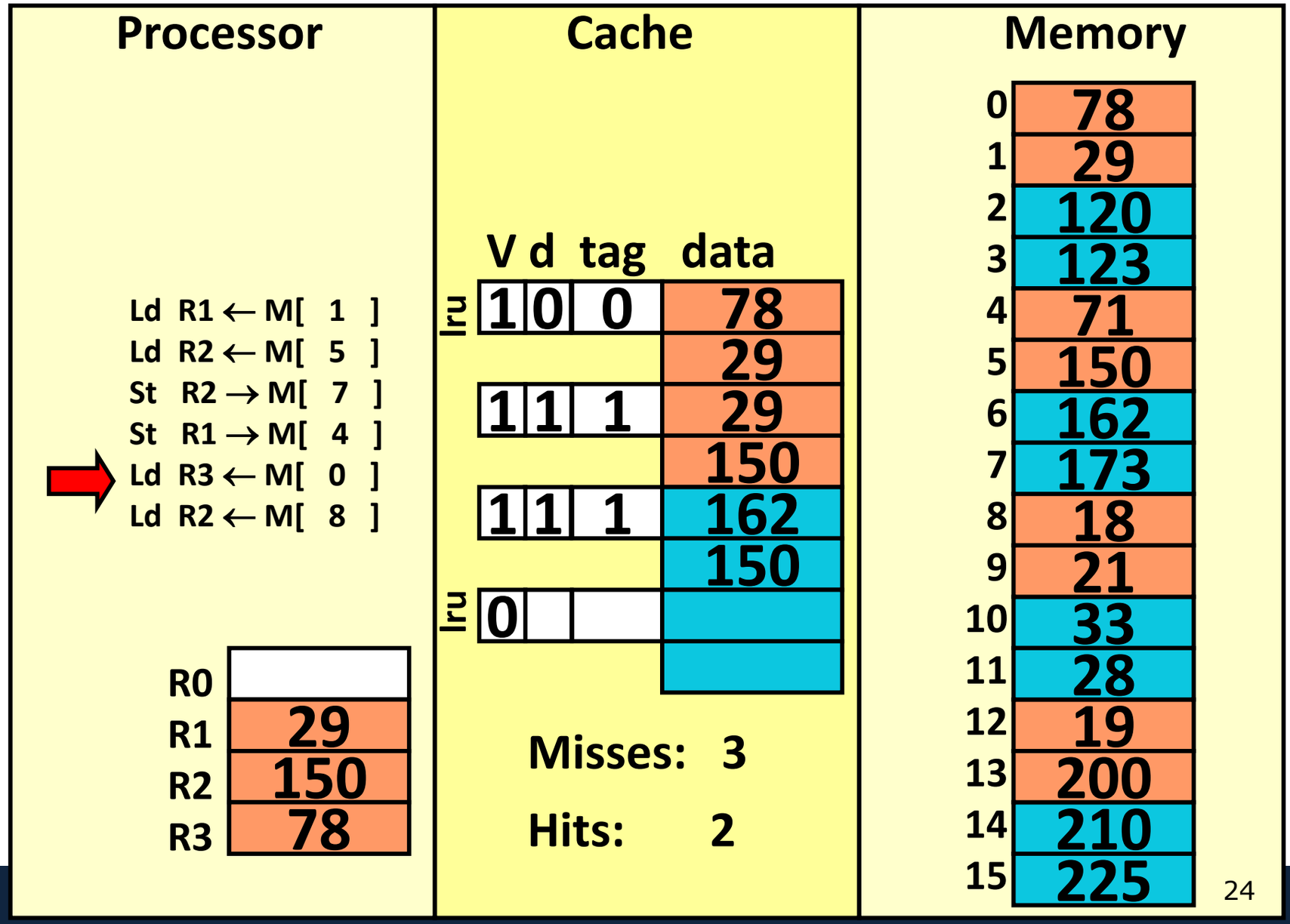
Set-associative cache (REF 4)



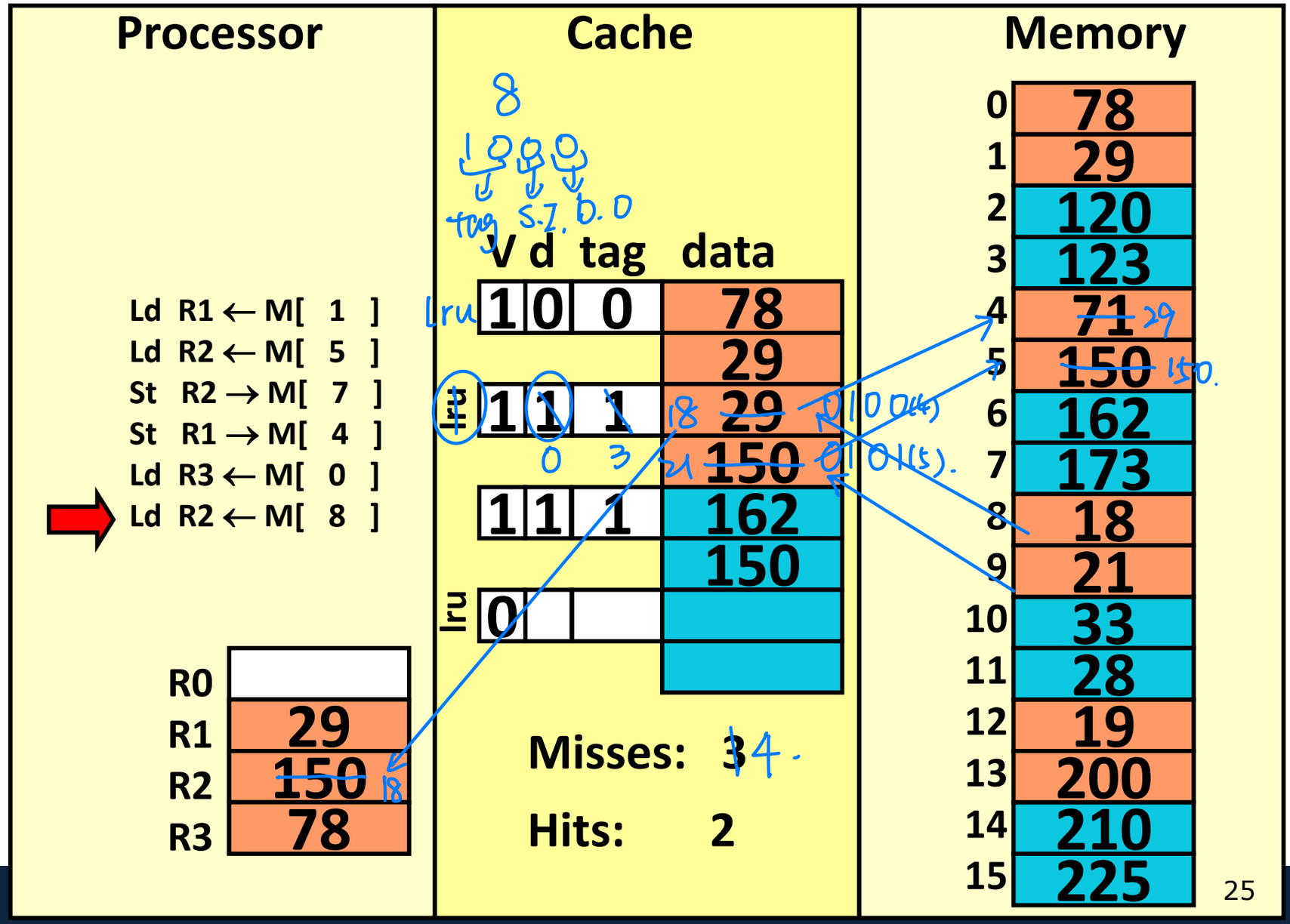
Set-associative cache (REF 5)



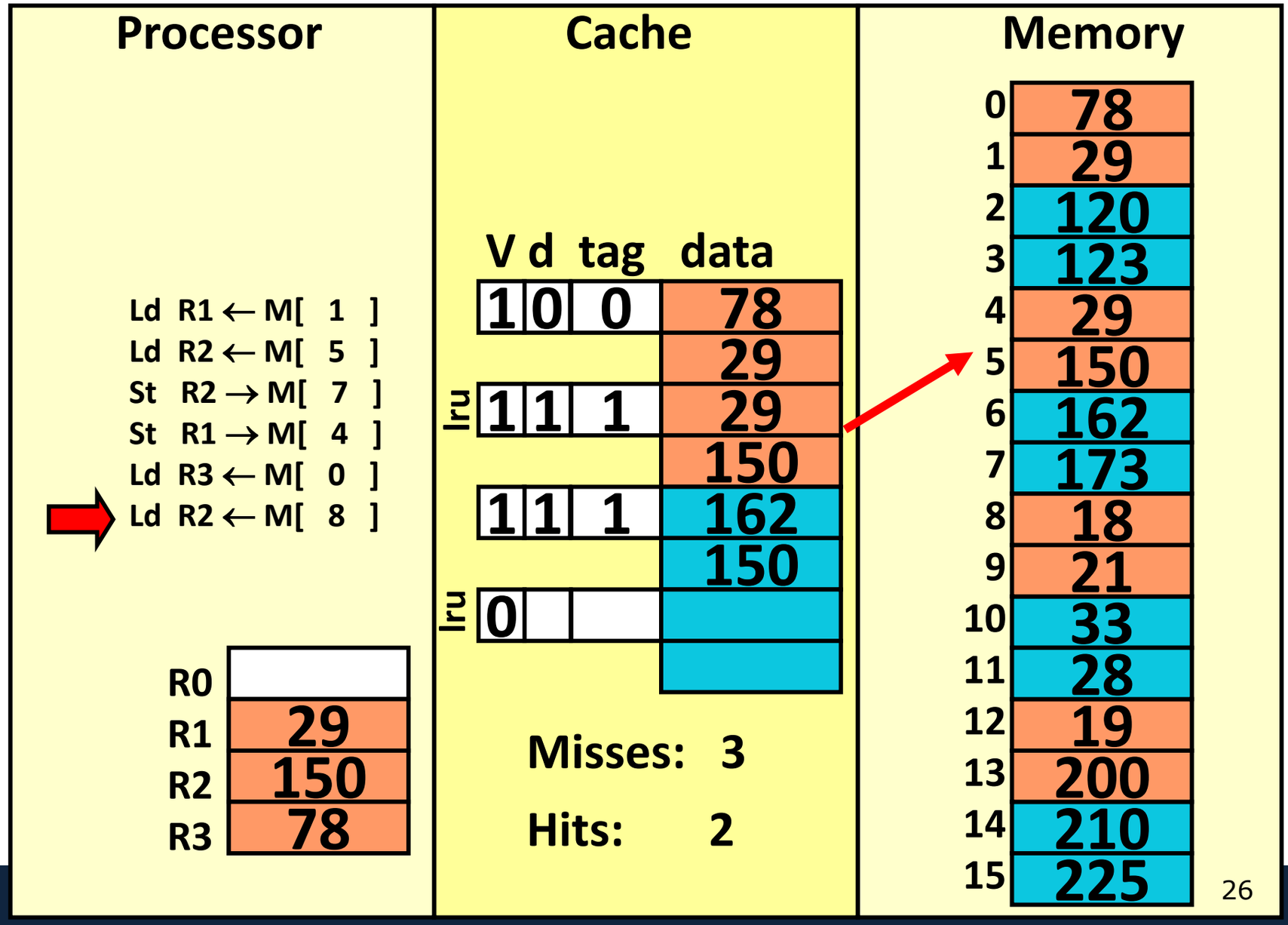
Set-associative cache (REF 5)



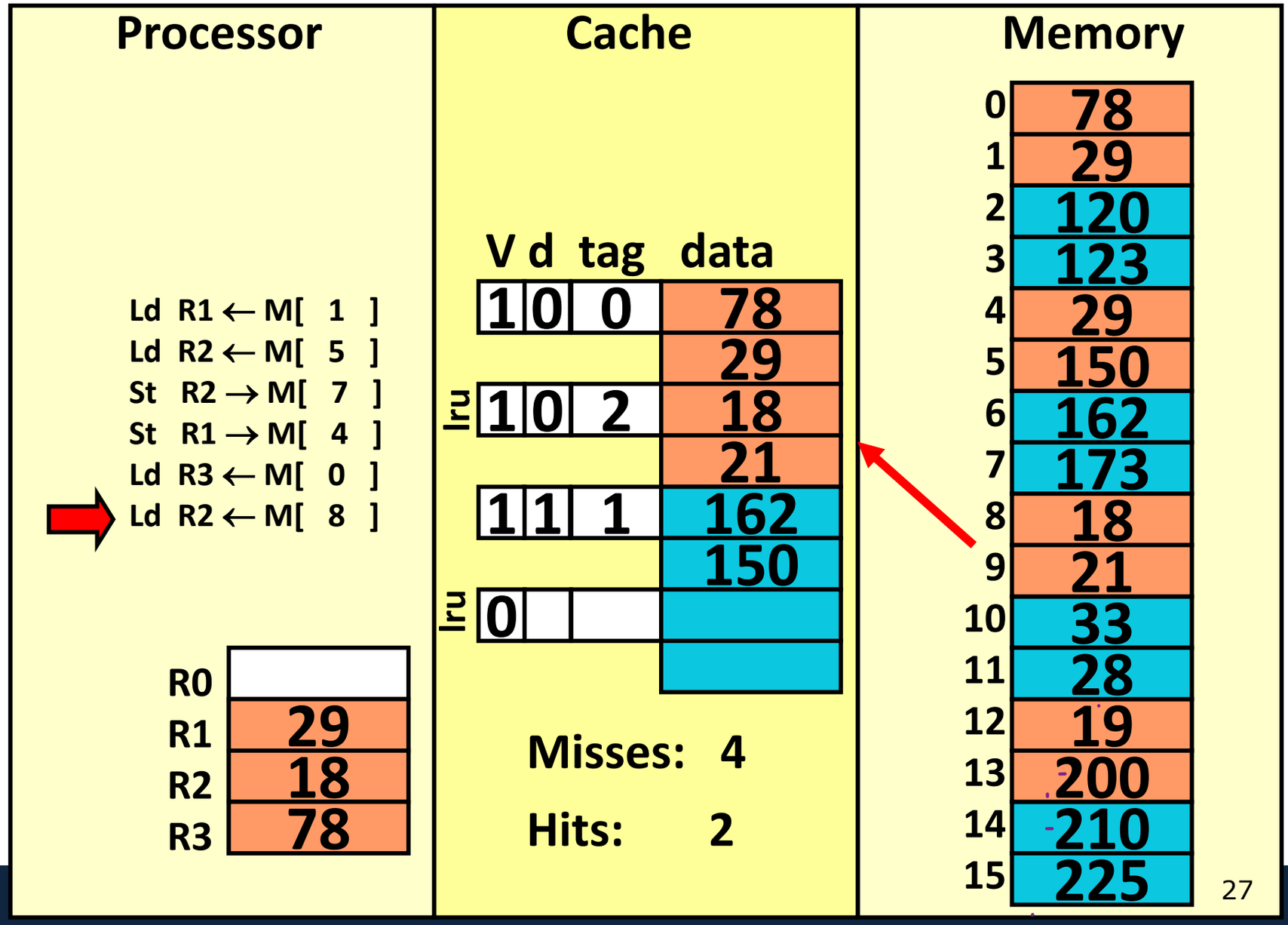
Set-associative cache (REF 6)



Set-associative cache (REF 6)



Set-associative cache (REF 6)



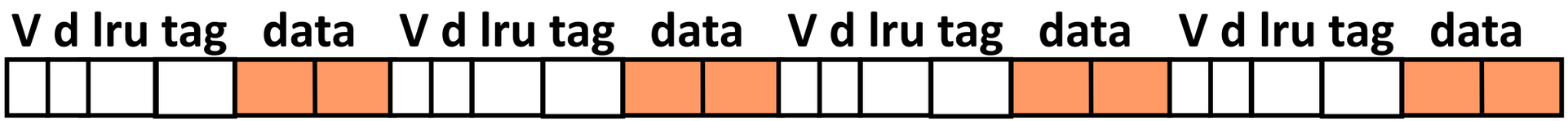
Agenda

- Set-associativity overview
- Example
- **Class problem**
- Integrating caches into our processor

Cache Organization Comparison

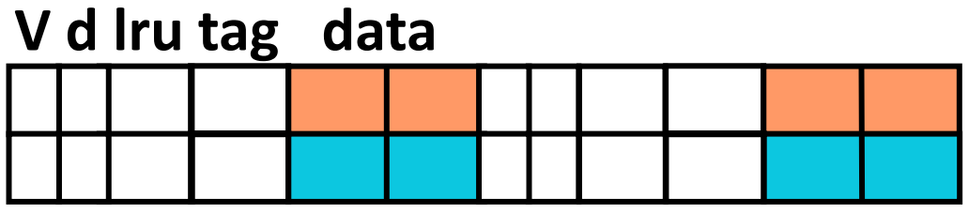
Block size = 2 bytes, total cache size = 8 bytes for all caches

1. Fully associative (4-way associative)

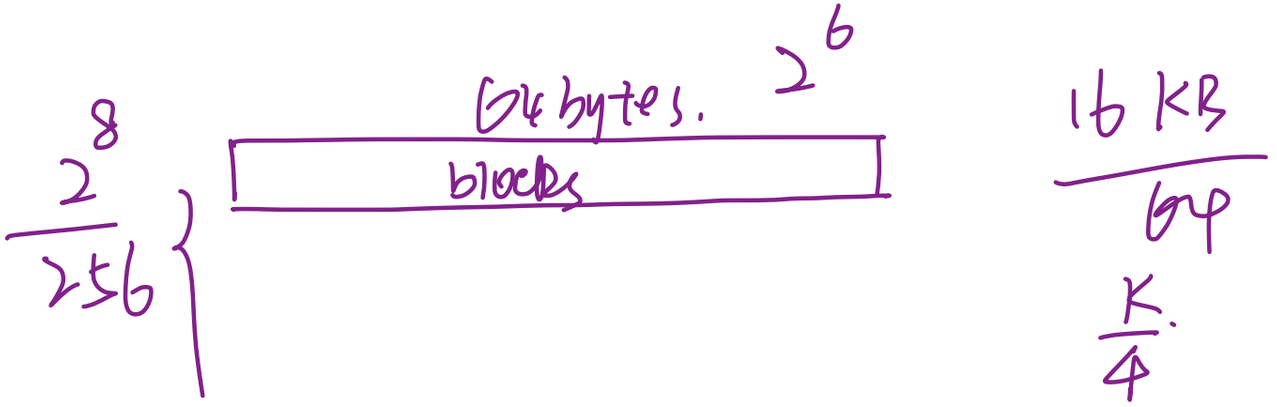


2. Direct mapped

3. 2-way associative



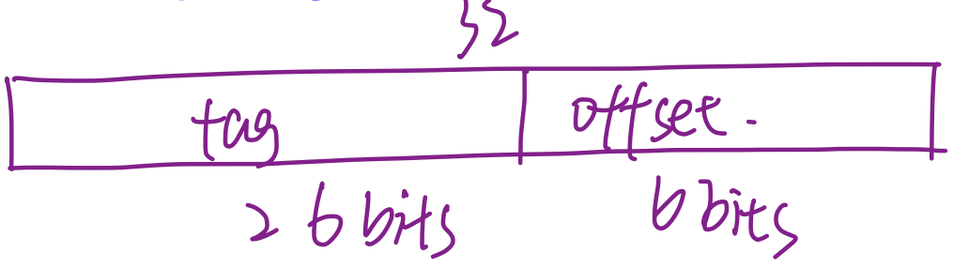
Class Problem 1



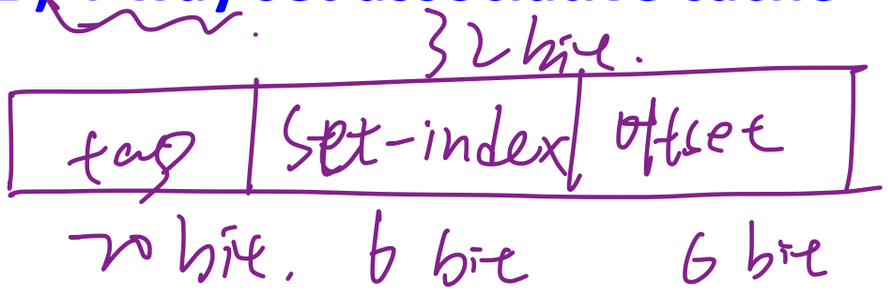
- For a 32-bit address and 16KB cache with 64-byte blocks, show the breakdown of the address for the following cache configuration:

$2^8 \div 2^2 = 2^6$

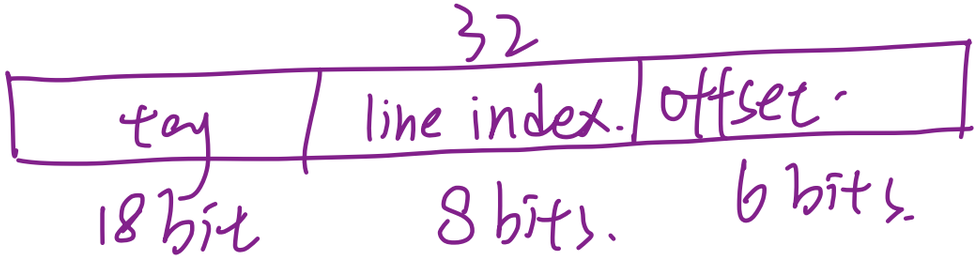
A) fully associative cache



B) 4-way set associative cache



C) Direct-mapped cache



Class Problem 1

16 kB ÷ 64 B = 1/4 k = 1024/4 = 256

64-byte blocks.

64 block in one cache.

- For a 32-bit address and 16KB cache with 64-byte blocks, show the breakdown of the address for the following cache configuration:

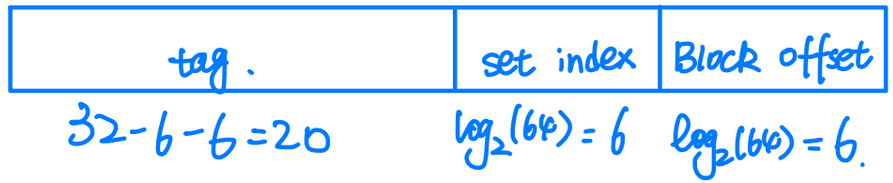
256 ÷ 4 = 64 64 sets each set has 4 ways

A) fully associative cache

memory address

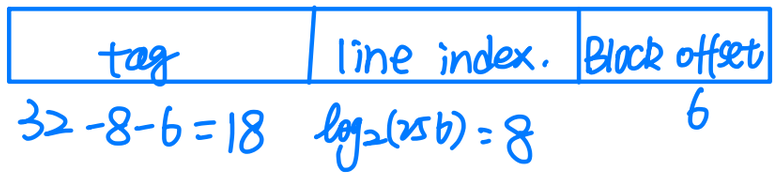
B) 4-way set associative cache

memory address



C) Direct-mapped cache

memory address.



Class Problem 1

- For a 32-bit address and 16KB cache with 64-byte blocks, show the breakdown of the address for the following cache configuration:

A) fully associative cache

Block Offset = log2(64)=6 bits
Tag = 32 - 6 = 26 bits

C) Direct-mapped cache

Block Offset = 6 bits
#lines = 256 Line Index = 8 bits
Tag = 32 – 6 – 8 = 18 bits

B) 4-way set associative cache

Block Offset = 6 bits
#sets = #lines / ways = 64
Set Index = 6 bits
Tag = 32 - 6 - 6 = 20 bits

Agenda

- Set-associativity overview
- Example
- Class problem
- **Integrating caches into our processor**

Multi-Level Caches

- We've been considering proc -> cache -> memory

- This works well if working data set is <= size of cache

- But if data set is a little larger that cache, performance can plummet

Multi-Level Caches

- This is the motivation of multiple levels of caches
- L1 – smallest, fastest, closest to processor
- LN – biggest, slowest, closest to memory
- Allows for gradual performance degradation as data set size increases
- 3 levels of cache is pretty common in today's systems

What about cache for instructions

- We've been focusing on caching loads and stores (i.e. data)
- Instructions should be cached as well
- We have two choices:
  1. Treat instruction fetches as normal data and allocate cache lines when fetched
  2. Create a second cache (called the instruction cache or ICache) which caches instructions only
     - More common in practice

How do you know which cache to use?

What are advantages of a separate ICache?

Integrating Caches into Pipeline

- How are caches integrated into a pipelined implementation?
  - Replace instruction memory with Icache
  - Replace data memory with Dcache

- Issues:
  - Memory accesses now have variable latency
  - Both caches may miss at the same time

Next time

- How to properly choose cache parameters?
  - Start by classifying why misses occur

Extra Slides

Improving our Caches

- If our cache is getting a lot of misses, how do we improve it?
  - Depends on why the misses occurring
  - Is the cache too small? Is the associativity too restrictive? Something else?
- A decent first step is to **classify** the types of missing we are observing

Classifying Cache Misses

- Cache misses happen for 3* reasons
  - The 3C's of Cache misses:
- Compulsory miss
  - We've never accessed this data before
- Capacity miss
  - Cache is not large enough to hold all the data
  - May have been avoided if we used a bigger cache
- Conflict miss
  - Cache is large enough to hold data, but was replaced due to overly restrictive associativity
  - May have been avoided if we used a higher-associative cache

**On multi-core systems, there's a 4th C – take EECS 470/570 to learn more*

Classifying Cache Misses

- Scenario: run given program on system with N-way cache of size M
  - Identify each miss
- We can classify each miss in a program by simulating on 3 different caches
  - If miss still occurs in cache where size >= memory size: compulsory miss
  - Else, if miss occurs in fully associative cache of size M: capacity miss
  - Else, if miss occurs in N-way cache of size M (original cache): conflict miss

Agenda

- Motivation
- **Example**
- How to optimize cache design
- Practice Problem 1
- Practice Problem 2
- Practice Problem 3
- Practice Problem 4

3C's Sample Problem

Consider a cache with the following configuration: write-allocate, total size is 64 bytes, block size is 16 bytes, and 2-way associative. The memory address size is 16 bits and byte-addressable. The replacement policy is LRU. The cache is empty at the start.

For the following memory accesses, indicate whether the reference is a hit or miss, and the type of a miss (compulsory, conflict, capacity)

3 C's Practice Problem – 3 C's

64 bytes total, 16 byte blocks, 2-way, 2 sets

Address	Infinite	FA	SA	3Cs
0x00				
0x14				
0x27				
0x08				
0x38				
0x4A				
0x18				
0x27				
0x0F				
0x40				

3 C's Practice Problem – 3 C's

Poll: How many blocks will be in a 64 byte FA cache?

Address	Infinite	FA	SA	3Cs
0x00	M			
0x14	M			
0x27	M			
0x08	H			
0x38	M			
0x4A	M			
0x18	H			
0x27	H			
0x0F	H			
0x40	H			

3 C's Practice Problem – 3 C's

64 bytes total, 16 byte blocks, 2-way, 2 sets

Address	Infinite	FA	SA	3Cs
0x00	M	M		
0x14	M	M		
0x27	M	M		
0x08	H	H		
0x38	M	M		
0x4A	M	M		
0x18	H	M		
0x27	H	M		
0x0F	H	M		
0x40	H	H		

3 C's Practice Problem – 3 C's

64 bytes total, 16 byte blocks, 2-way, 2 sets

Address	Infinite	FA	SA	3Cs
0x00	M	M	M	Compulsory
0x14	M	M	M	Compulsory
0x27	M	M	M	Compulsory
0x08	H	H	H	---
0x38	M	M	M	Compulsory
0x4A	M	M	M	Compulsory
0x18	H	M	H	---
0x27	H	M	M	Capacity
0x0F	H	M	M	Capacity
0x40	H	H	M	Conflict